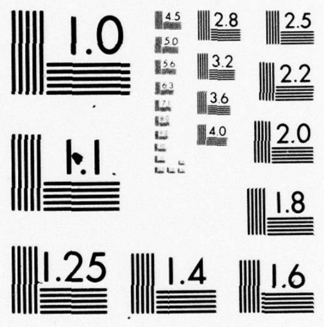
1 OF 1
ADA049 553

END
DATE
FILMED
3-78
DDC


1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

(4)

AFOSR-TR- 77 - 1293

COLLEGE of BUSINESS and ADMINISTRATION

Wright State University
dayton, ohio

(4)

(6) # MULTI-ITEM SCHEDULING IN REPARABLE ITEM INVENTORY SYSTEMS WITH REFLECTION PROGRAMMING.

(9) Interim rept.,

by

(10) W. Steven Demmy

(11) Apr 77

(15) AFOSR-76-3011

(12) 39p.

(16) 2304 (17) A5 April, 1977

(14) WP-

Working Paper 76-3011-4
Department of Administrative Science and Finance
Wright State University
Dayton, Ohio 45431

(18) AFOSR

(19) TR-77-1293

New 410555

mt

MULTI-ITEM SCHEDULING IN REPARABLE ITEM INVENTORY SYSTEMS WITH REFLECTION PROGRAMMING

by

W. Steven Demmy
Assistant Professor of Quantitative Business Analysis
Department of Administrative Sciences and Finance
Wright State University

The repair and overhaul of military equipment is a costly and complex logistics activity. Factors that may have significant impact upon scheduling decisions in such systems include costs of set-up, production, and expediting, obsolescence probabilities, the availability of the reparable assets, manpower, equipment and funds required to support the repair effort. This paper reviews the current state-of-the-art for solving multi-item repair scheduling problems in such systems, and presents an algorithm for solving these problems using Generalized Upper Bounding with dynamic programming subproblems. Computational experience involving up to 170 items, 48 constraints, and 12 planning periods is also reported.

TABLE OF CONTENTS

Multi-Item Scheduling In
Reparable Item Inventory Systems
With Reflection Programming

I. Introduction

The depot level repair and overhaul of military equipment is an important and costly logistics activity. In the Air Force Logistics Command alone, depot maintenance support accounts for the work of about 40,000 persons and requires annual expenditures of over a billion dollars.

An important aspect of depot maintenance is the development of repair schedules and manpower plans which result in the efficient use of reources while providing high levels of support to customers of the inventory/repair system. Ideally, the development of such plans requires simultaneous consideration of the magnitude, timephasing, and priority of alternative repair programs and the availability and cost of the skills, equipment, facilities, and funds required to support these programs. This implies the need to collect and manipulate large masses of data and to make a number of trade-off decisions to obtain efficient use of available resources.

Until recently, the very large number of variables associated with depot repair planning and the fragmented nature of "second generation" data systems have generally precluded the use of mathematical programming techniques to assist in the repair planning process. The advent of powerful new techniques for solving large scale optimization problems, the trend toward integrated data systems, and the availability of modern computers with tremendous memory and computing power indicate these problems could diminish for many repair organizations during the current decade.

In this paper we present a method for solving repair scheduling and manpower allocation problems using large scale optimization techniques. The paper has five sections. Section I provides background on the nature of the repair scheduling problem and reviews the current state of the art in this area. Section II presents a deterministic mixed integer model of the repair scheduling problem. Section III discusses how linear programming might be applied to obtain approximate solutions to the original integer optimization problem, while Section IV discusses how reflection programming may be applied to obtain heuristic solutions to this problem. Finally, Section V presents computational results for the latter two solution methods.

Background

The development of depot repair plans involves two basic categories of decisions:

(a) Workloading, which determines the specific items to be repaired, the quantities involved, and when and where the repair is to be performed.

(b) Resource allocation, which determines how available resources should be adjusted and allocated to support assigned repair workloads.

In reaching the above decisions, it is necessary to strike a good balance between two conflicting objectives: providing high levels of support to customers and efficient use of available resources.

Figure II-1 shows some of the major factors that may have important influences in the development of cost-effective depot repair plans. As illustrated in the figure, several constraints usually must be satisfied in developing meaningful repair plans and programs. Limitations on the availability of skills, equipment, facilities, and funds may preclude doing all desired repair programs within specified time limits, while legal, administrative, or policy constraints established by higher authorities, e.g., Congress or the

FACTORS

Constraints

.overtime
.manpower
.equipment
.facilities
.funding

.legal
.administrative
.policy
.political
.lead times to obtain resources

Economic Factors

.personnel
  .wages, overtime
  .hiring, layoff, training, transfers
  .attrition

.equipment and facilities
  .maintenance
  .modernization
  .expansion

.producitivity effects
  .setup and calibration time
  .learning curve effects
  .inventory carrying costs
  .efficiency of scale

.alternate sources of repair

Material Availability

.reparable assets
.subassemblies
.bits and pieces
.replacement factors

Time Period

.today
.this week
.this month
.this quarter
.this year
.next five years

Data System Factors

.data collection
.storage
.computation
.errors

Priority of Repair Requirements

Figure I-1 Factors

Executive Branch, may further restrict the range of alternatives available to repair planners. Additonally, the limited availability of reparable assets, subassemblies, and "bits and pieces" needed in the repair process may also make certain work-loading decisions infeasible or uneconomical.

Economic factors that may have an important bearing on the development of repair plans include workforce wage rates, costs of manpower adjustments (over-time, hiring, layoff, training, transfers), and costs for the maintenance, modernization or augmentation of existing equipment and facilities. Other economic factors having important impacts in certain circumstances include times required for the setup and testing of complex repair equipment, learning curve effects, and the availability of alternative sources of repair, e.g., qualified subcontractors. Support effectiveness considerations include the essentiality of the work to be done and the uncertainty associated with the repair requirement.

The planning horizon under consideration may also have an important impact. Plans that are to be implemented within the next month require far more detailed scrutiny than plans that are to be implemented in future fiscal years. Further, less uncertainty is generally associated with them.

Another major factor is the organizational momentum created by the planning process itself. In most organizations, intermediate and long-range plans provide the basis for contractual agreements with suppliers and subcontracts and for carrying out of long-term personnel training programs. Once established, such agreements and programs are usually difficult and expensive to modify.

Current State of Knowledge

Studies to determine economic order and repair quantities for single item repair systems have been reported by several authors. These include the work of Allen and D'Esspo (1968), Brown (1969), Demmy (1970), Freiheit (1967), Hastings (1967), Krause (1970), Schrady (1967), Sherbrooke (1966), Sherbrooke

(1968), Simon (1969), Porteous and Landsdowne (1973), Faucett (1969), Muckstadt (1973), and Faucett (1972). Several other studies aimed at applying Sherbrooke's METRIC model to Air Force requirements computations have been completed by the Management Sciences Office (AFLC/XRS), Headquarters Air Force Logistics Command, Wright Paterson Air Force Base, Ohio. These studies provide results for computing spares acquisition quantities, or for scheduling in single item systems; however, they do not provide techniques for scheduling item repair when several resources (e.g. manpower, funds, or test equipment) limit scheduling flexibility.

Another important line of research is the development of techniques for solving multi-item scheduling problems in manufacturing industries. The problem of scheduling the production of many items over time was first studied via linear programming by A. S. Manne (1958). Additional work along these lines has been done by Dzielinski, Baker, and Manne (1963), Dzielinski and Gomory (1964), Kortanek, Sodaro, and Soyster (1968), Gornstein (1970), Lasdon and Terjung (1971), Demmy (1973), Demmy and Brock (1974), Demmy (1975), and Gornstein (1976). Dzielinski and Gomory applied the Dantzig-Wolfe Decomposition Principle to a problem very similar to that formulated by Manne (1958). The resulting linear program has a very large number of variables and possibly many constraints. Application of the decomposition principle yields an equivalent linear program, called the master program, with fewer constraints but even more variables. However, these may be dealt with by a column-generation technique, using sub-problems. For the problem studied, these problems were of the Wager-Whittin (1958) type and may thus be solved efficiently. Gornstein (1976) describes how this method was implemented as the basis for production planning in a major stamping plant.

Although the decomposition algorithm of Dzielinski and Gomory is an attractive approach, difficulties still remain. The most serious of these is that the linear program being decomposed is only an approximation to an integer programming problem whose solution is actually desired, and is a good approximation only when the number of items is much greater than the number of time periods. Under these conditions basic solutions to the linear program have the property that most of the variables will have integer values. This property is not shared by the master program of the decomposition principle. Thus the Dantzig-Wolfe decomposition approach cannot guarantee a majority of integer-valued variables until optimality. A second problem is that the reformulated problem may have a tremendous number of near optimal solutions prior to optimality, leading to extremely long computing times in arriving at an optimal solution.

Lasdon and Terjung (1971) proposed that the linear program formulation of Manne be attacked directly. In their approach, the large number of columns is handled by column generation via sub-problems. The multiplicity of constraints is dealt with by using Generalized Upper Bounding (GUB) to reduce the effective basis to a much smaller size. Although the method does not guarantee integer solutions, the round-off problem is much less severe in the practical problems reported. An extended version of the Lasdon-Terjung model forms the core of an automated production-scheduling and inventory-control system currently being used by a major U.S. manufacturer.

Demmy and Brock (1974) present a general procedure for depot repair planning based upon a large-scale linear programming model of the repairs scheduling process. Factors considered in this general model include constraints on overtime, manpower, equipment, facilities, and funding. Economic factors considered include personnel costs related to wages, overtime, hiring, lay-off, training, transfers, and attrition, costs for equipment and facility maintenance, modernization, and expansion, producitivty effects related to set-up and calibration time, learning curve effects, inventory carrying costs, efficiencies of scale, and the possibilities of alternate sources of repair. Although the Demmy-Brock model is very general, it appears that currently existing large scale mathematical programming packages such as MPSX (IBM), APEX-2 (CDC), LP/6000 (Honeywell) may be used to solve the model for aggregate planning problems such as those faced by headquarters organizations. However, because of the very large number of variables associated with detailed scheduling problems, special techniques must be developed to solve these latter problems in a reasonable amount of computer time.

Demmy (1973) developed an algorithm for solving a simplified version of the Demmy-Brock model using the Dantzig-Wolfe decomposition procedure. Although computational experience with test problems involving up to 90 items, 12 constraints and 4 planning periods was encouraging, the computational difficulties related to the Dantzig-Wolfe decomposition formulation remained. Later, Demmy (1977) generalized this model to consider the joint determination of procurement and repair schedules under very general assuptions about the nature of the repair cost functions. Cost elements explicitly considered in the later model include set-up costs, learning curve effects, expediting and inventory holding costs, and penalties for incurring shortages. The model also includes provision for obsolescence costs and time-discounting effects.

As we shall see later, the repair shceduling problem considered in this paper is a special case of a fixed-charge integer programming problem with a special "block angular" structure. Hence, one approach to the repair scheduling problem is to use solution procedures designed for general fixed charge problems.

Exact solution methods for fixed charge problems may be classified into three categories: extreme point ranking procedures [Gray (1971), Murty (1968], branch and bound procedures [(Jones and Solard (1962) and Steinberg (1970)], and decomposition methods [(Kochman (1976)]. Unfortunately, these methods appear to have little practical value for large scale problems. Consequently, several heuristic approaches have been proposed that generate near-optimal solutions. Generally, these heuristics start by producing a good extreme point solution. Adjacent extreme points are then examined until a local minimum is determined. Then a move is made to an extreme point away from this local minimum, and the process is repeated until no further improvement is obtained or until a termination criteria (such as a specified number of interations) is met. Heuristics of this type are reported by Balinski (1961), Cooper and Drebes (1967), Denzler (1969), Roussean (1973), and Steinberg (1970).

A final line of research relative to this paper deals with recent developments in the mathematical programming literature. In particular, Agin and Cullen (1975) have developed a heuristic procedure called reflection programming which is a variant of the Dantzig-Wolfe decomposition principle. Agin and Cullen applied their technique to a problem of transport routing and vehicle loading. They report a number of computational advantages in applying reflection programming to this problem. These include:

a. Integer-valued solutions. The variables always satisfy integer restrictions.

b. Decomposition of the master program. Due to the special structure of the master program, a "reflection pivot" can be implemented without computing the inverse of the master problem basis matrix. This results in substantial reductions in both computational requirements and computer core memory.

c. Rapid Solution. Agin and Cullen noted that their reflection programming algorithm terminates computation much more rapidly than the decomposition algorithm.

Although reflection programming does not guarantee optimal solutions, Agin and Cullen imply that very good vehicle routing solutions are achieved using the technique though no data supporting this contention is given. They report that truck routing problems involving 7200 equations in the master problem, 6000 variables per sub-problems and 31 sub-problems were solved using an IBM 360/65 in between 1 and 2 minutes of CPU time including time for set-up and reports. Although the reparable item scheduling problem has a different structure than that considered by Agin and Cullen, it appears that reflection programming concepts are applicable to the latter problem.

This paper presents the mathematical details involved in applying reflection programming to the repair scheduling problem, and presents computational experience with the method.

II. A Deterministic Model

Let us now develop a model that captures many of the factors described in the previous section. We shall use the following symbols:

A. Indices

$i = 1,2,\ldots,I$ denotes the time period under consideration

$j = 1,2,\ldots,J$ denotes a specific item to be repaired

$k = 1,2,\ldots,K$ denotes a specific resource required in the repair process. We shall use the index value $k=0$ to denote cost elements that are part of the objective function.

B. Status Variables

v_{ij} = quantity of reparable units of item j on hand at the end of period i

w_{ij} = quantity of serviceable units of item j on hand at the end of period i.

S_{ij} = slack quantity of resource k in period i; i.e. the quantity of resource k with no planned utilization in period i.

C. Model Inputs

1. item data

d_{ij} = demand for servicable units of item j during period i

e_{ij} = number of reparable units of item j reaching the repair facility during period i.

v^*_{1j} = number of reparable assets on-hand at the beginning of period 1

w^*_{1j} = number of servicable assets on-hand at the beginning of period 1. Negative values for w_{1j} denote initial period backorders.

h_{ij} = cost of having one servicable unit of item j on hand at the beginning of period i

a_{kj} = amount of resource k required to "set-up" and begin repair of item j.

b_{kj} = amount of resource k required to produce one unit of resource k, given the job has already been set-up.

2. Resource data

b_{ik} = amount of resource k available during period i

b^o_{ik} = maximum additional hours of resource k that can be made available in period i by working overtime.

R^s_{ik} = cost per unit slack of resource i in period k, i.e. the opportunity cost associated with each unit of available capacity b_{ik} that is not utilized in period i.

R^o_{ik} = cost per unit of overtime associated with period i and resource k.

R^A_{ik} = cost per unit of resource k that is subcontracted during period i. In this paper, we assumed $R^o_{ik} < R^A_{ik}$.

D. Decision Variables

q_{ij} = quantity (in units) of item j scheduled for repair in period i

O_{ik} = number of hours of resource k to be provided by overtime in period i.

A_{ik} = number of hours of resource k to be provided from auxiliary sources (such as subcontracting) during period i

For simplicity, we assume

a. Repair may be completed in a single period

b. One reparable asset is required at the beginning of a period for each serviceable unit scheduled for completion during that period. (i.e. we assume no items are scrapped during the repair process).

c. Both demand d_{ij} and reparable item deliveries e_{ij} are known deterministically.

d. Backorders are not permitted.

For treatment of a more complex situation involving multi-period lead times, probabilistic demand and reparable generations, and general cost functions, see Demmy (1975). For this latter situation, the optimization method defined below is still appropriate, but the notation is much more complex.

Problem I. The Multi-Item Scheduling Problem

We wish to determine values of q_{ij}, O_{ij} and A_{ik} that minimize the total costs of repair, holding inventory and adjusting resource levels during the planning horizon. In symbols, we wish to minimize Z,

(1) $$Z = \Sigma_{ij} [a_{oj} \delta (q_{ij}) + b_{oj} q_{ij}] + h_{ij} w_{ij} + \Sigma_{ik} [R^{S}_{ik} S_{ik} + R^{O}_{ik} O_{ik} + R^{A}_{ik} A_{ik}]$$

In solving this problem, the decision variables must satisfy the following constraints:

Resources Required Must Equal Resources Available

(2)

(2) $$\Sigma_{j} [a_{kj} (q_{ij}) + b_{kj} q_{ij}] + S_{ik} - O_{ik} - A_{ik} = b_{ik} \quad \text{for all } ik$$

The Overtime limit cannot Be Exceeded

(3) $$O_{ik} \leq b^{o}_{ik} \quad \text{for all ik}$$

Initial On-Hand Reparable and Servicable Assets Are Known

(4) $$v_{1j} = v^{*}_{1j} \quad \text{for all j}$$

(5) $$w_{1j} = w^{*}_{1j}$$

Reparable and Servicable Material Balance Constraints

(6) $$v_{i+1,j} = v_{ij} - q_{ij} + e_{ij} \quad \text{for all ij}$$

(7) $$w_{i+1,j} = w_{ij} + q_{ij} - d_{ij}$$

There must be Enough Reparable Assets to Support Scheduled Repairs

(8) $v_{ij} \geq q_{ij}$ for all ij

Non-Negativity Conditions

(9) $q_{ij}, w_{ij} \geq 0$ for all ij

$S_{ik}, O_{ik}, A_{ik} \geq 0$ for all ik

In addition, q_{ij} must be integer. This problem is a mixed integer programming problem with I(J+3K) variables, and I(2K+2J) constraints plus the non-negativeity constraints (9).

Observe that (4), (6), and (8) imply that

(10) $$v_{ij} = v^*_{ij} + \Sigma_{t=1}^{i-1} e_{tj} - \Sigma_{t=1}^{i-1} q_{tj} \geq q_{ij}$$

which may be rearranged to

(11) $$\Sigma_{t=1}^{i} q_{tj} \leq v^*_{ij} + \Sigma_{t=1}^{i-1} e_{tj}$$

In addition observe that (5), (7), and (9) imply that

(12) $$w_{i+1,j} = w^*_{ij} + \Sigma_{t=1}^{i} q_{tj} - \Sigma_{t=1}^{i} d_{tj} \geq 0$$

Together, these constraints place upper and lower bounds on the cummulative scheduled repair, specifically,

(13) $$\max[0, \Sigma_{t=1}^{i} d_{tj} - w^*_{ij}] \leq \Sigma_{t=1}^{i} q_{tj} \leq v^*_{ij} + \Sigma_{t=1}^{i-1} e_{tj}$$

The lower bound in (13) simply states that the cummulative repair must be greater than or equal to cummulative demands for serviceable items, net of any initial on-hand stocks. The upper bound in (13) states that the cummulative repair cannot exceed the cummulative reparable assets available at the beginning of each period. This upper bound defines the maximum production of item j that may occur by a given period i. Cummulative net demand in excess of this value cannot be supported for lack of reparable carcasses. In this paper, we assume that condition (13) is always satisfied. In practice, this may be accomplished by rescheduling some demand to later time periods, or by satisfying some demand through procurement actions rather than by repair.

III. Approximate Solution Using Linear Programming

The above model is very similar to capacitated lot size models studied by Manne (1958), Dzielinski, Baker, and Manne (1963), Dzielinski and Gomory (1965), and Lasdon and Terjung (1971). The new feature in this model is the addition of reparable asset constraints (4), (6), and (8).

Manne suggested that approximate solutions to capacitated lot size models might be obtained using linear programming by reformulating problem (1)-(9) as follows: Suppose we were to enumerate all feasible repair schedules for each item j, i.e. all schedules satisfying constraints (4)-(9). Let $Q_j^t = (q_{1j}^t, q_{2j}^t, \ldots, q_{Ij}^t)$ denote the tth schedule for item j. Further, let

(14) $$p_j^t = \Sigma_i [a_{oj}\delta(q_{ij}^t) + b_{oj}q_{ij}^t + h_{ij}w_{ij}^t]$$

(15) $$g_{ikj}^t = a_{kj}\delta(q_{ij}^t) + b_{kj}\, q_{ij}^t$$

Now let x_j^t be a (0,1) - integer variable such that $x_j^t = 1$ indicates schedule t is to be adopted for item j, and $x_j^t = 0$ denotes otherwise. Then the problem (1)-(9) is equivalent to the problem:

<u>Problem II</u>

(16) $$\text{Min } Z = \Sigma_{jt} p_j^t\, x_j^t + \Sigma_{ik}[R_{ik}^S S_{ik} + R_{ik}^O O_{ik} + R_{ik}^A A_{ik}]$$

subject to

(17) $$\Sigma_j \Sigma_t\, g_{ikj}^t x_j^t + S_{ik} - O_{ik} - A_{ik} = b_{ik} \qquad \text{all } i,k$$

(18) $$O_{ik} \leq b_{ik}^o \qquad \text{all } i,k$$

(19) $$\Sigma_t \quad x_j^t = 1 \qquad \text{all } j$$

(20) $$S_{ik}, O_{ik}, A_{ik} \geq 0$$

where x_j^t is integer.

We shall refer to problem (16)-(20) as Problem II. Observe that if the x_j^t-integer requirement is replaced by the constraint

(21) $x_j^t \geq 0$ for all j^t

then problem II is a linear programming problem in the variables x_j^t, S_{ik}, O_{ik}, and A_{ik}. This problem has (2IK+J) - constraints and a very large number of decision variables. For most real-world situations, Problem II is still too large to solve using conventional techniques, but by exploiting its special structure, both computer memory and calculation requirements may be significantly reduced.

Lasdon and Terjung (1971) observed that the computational burden to solve LP-problems such as (16)-(21) may be reduced by using

a. delayed column generation procedures to identify x_j^t - variables that are candidates to enter the basis.

b. generalized upper bounding (GUB) methods to reduce memory requirements for storage of the inverse of the basis to (16)-(21).

c. multiple pricing procedures to improve the rate of convergence toward the optimal solution.

Lasdon and Terjung's procedures for GUB and multiple pricing directly apply to (16)-(21), and will not be discussed further here. However, their methods must be modified to apply delayed column generation to the repair scheduling problem.

To see how this is done, let C_j^t denote the marginal cost associated with variable x_j^t,

(39) $C_j^t = p_j^t - \Sigma_{ik} \pi_{ik} g_{ikj}^t - \alpha_j$

where π_{ik} and α_j denote dual variables associated with (17) and (19), respectively. Using the standard simplex column selection rule, if there are several schedules with negative marginal costs, we will select the one with the most negative C_j^t. Hence, for a given item j, we seek to identify the particular schedule t* such that

(40) $$C_j^{t*} = \min_t[C_j^t] = \operatorname{Min}_t[p_j^t - \Sigma_{ik}\pi_{ik}q_{ijk}^t - \alpha_j]$$

Note, however, that relation (40) is equivalent to the single item problem of minimizing Z_j,

(41) $$Z_j = \Sigma_i[a_{oj}\delta(q_{ij}^t) + b_{oj}q_{ij}^t] - \Sigma_{ik}\pi_{ik}[a_{kj}\delta(q_{ij}) + b_{kj}q_{ij}] - \alpha_j$$

(42) $$= \Sigma_i\ [(a_{oj} + \Sigma_k\pi_{ik}a_{kj})\ \delta(q_{ij}^t) + (b_{oj} + \Sigma_k\pi_{ik}b_{ij})q_{ij}^t + h_{ij}w_{ij}^t] - \alpha_j$$

(43) $$= \Sigma_i\ \bar{a}_i\delta(q_{ij}^t) + \bar{b}q_{ij}^t + h_{ij}w_{ij}^t - \alpha_j$$

subject to the constraints

(44) $$v_{ij} = v_{ij}^*$$

(45) $$w_{ij} = w_{ij}^*$$

(46) $$v_{i+1,j} = v_{ij} - q_{ij} + e_{ij}$$

(47) $$w_{i+1,j} = w_{i+1,j} + q_{ij} - d_{ij}$$

(48) $$q_{ij} \leq v_{ij}$$

(49) $$q_{ij},\ v_{ij}, w_{ij} \geq 0$$

In general, the subproblem (43)-(49) may be solved as a dynamic programming problem with one state variable. To see this recall that (46) and (47) may be written as:

(50) $$v_{i+1,j} = v_{1j}^* + \Sigma_{t=1}^{i}d_{ij} - \Sigma_{t=1}^{i}q_{ij}$$

(51) $$w_{i+1,j} = w_{ij}^* - \Sigma_{t=1}^{i}d_{ij} + \Sigma_{t=1}^{i}q_{ij}$$

Let us define r_{ij} = cummulative units of item j repaired in periods <u>preceding</u> period i, i.e.

(52) $r_{1j} = 0$

and

(53) $r_{ij} = \Sigma_{t=1}^{i-1} q_{ij}$ for $i=2,3,\ldots,I$

Now, let

(54) $Z_{I+1}(r) = 0$ for all r

and

(55) $Z_i(r)$ = minimum cost in periods $i, i+1,\ldots,I$ if r units have been repaired in periods preceding period i, and an optimal policy is followed in period i and succeeding periods.

Values for $Z_i(r)$ may be determined by backward recursion by evaluating

(56) $$Z_i(r) = \underset{0 < q < v_{ij} - r}{\text{Min}} \quad [\bar{a}_i\delta(q_{ij}) + \bar{b}_i q + h_{i+1,j}(w^*_{ij}+r+q) + Z_{i+1}(r+q)]$$

where

(57) $v^*_{ij} = v^*_{1j} + \Sigma_{t=1}^{i-1} e_{ij}$

for $i=2,3,\ldots,I$

(58) $w^*_{ij} = w^*_{ij} - \Sigma_{t=1}^{i} d_{ij}$

Feasible values for r are bounded by (13). In terms of our current symbols,

(59) $\max\, [0,-w^*_{i-1,}j] \leq r \leq v^*_{i-1,}j$

Hence, it is only necessary to evaluate (56) for integer values of r satisfying (59). At the completion of the recursion calculations, the value $Z_o(0)$ is the minimum value of Z_j in (41); the associated repair schedule may then be found by a standard backtracking calculation.

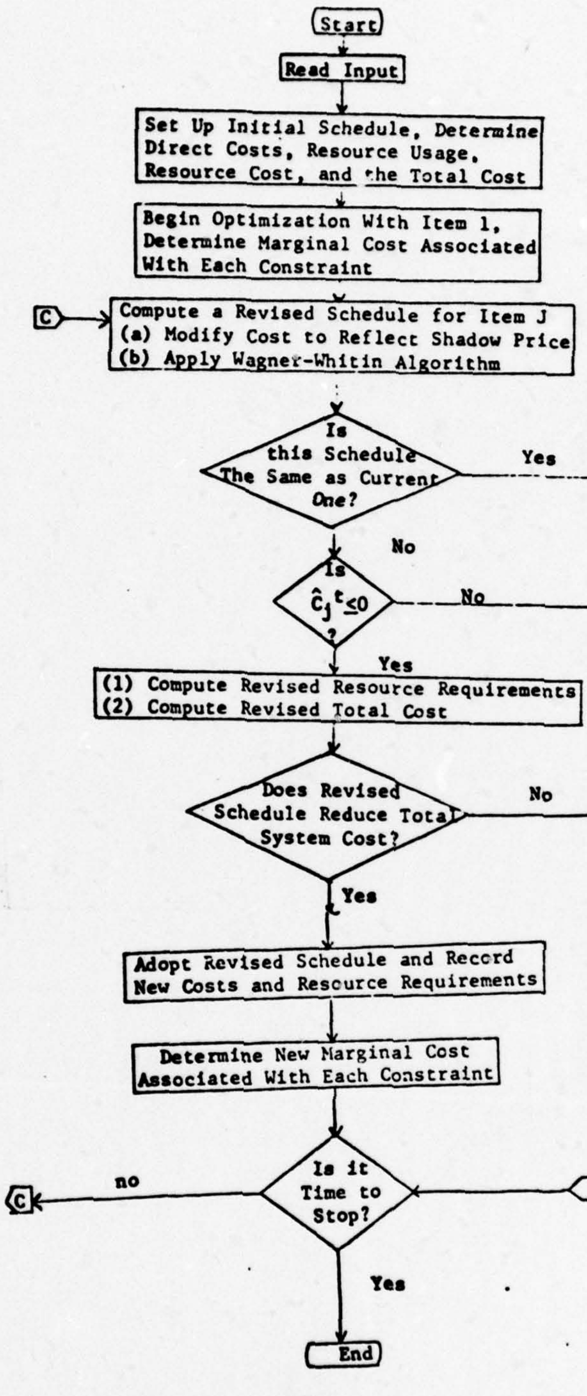
Section IV. Reflection Programming

As noted above, Agin and Cullen (1975) have developed a heuristic procedure called reflection programming which is a variant of the Dantzig-Wolfe decomposition principle. They report a number of computational advantages of their method in applying reflection programming to problems of transport routing and vehicle loading. These include rapid solution, small computer memory requirements, and the property that variables always satisfy integer restructions. In this section, we develop the mathematics required to apply reflection programming to multi-item repair scheduling problems. In the next section, we report computational experience with this method.

The major steps involved in applying reflection programming to the repair scheduling problem are outlined in Figure IV-1.

Basically, the algorithm works by requiring that at any given time there is one and only one non-zero X_j^t associated with each item j. Once an initial solution is established, the algorithm proceeds by considering each item in succession. When a given item is considered, shadow prices associated with each resource are used to determine a trial schedule Q_j* that represents a marginal improvement over the existing schedule Q_j for the current item-- the same as in Dantzig-Wolfe decomposition. However, reflection programming requires that an entering schedule must completely replace the previous schedule. This one-for-one replacement will generally require adjustments in the basis variables associated with constraints (17)-(18).

These adjustments are referred to as the reflection pivot, since the impact of the schedule change is being "reflected" into its corresponding resource adjustments. In general, exchanging the current schedule for the trial schedule will decrease resource costs, but will increase production and inventory costs-- particularly when demand is highly seasonal. If the total of these cost changes


Start
Read Input
Set Up Initial Schedule, Determine Direct Costs, Resource Usage, Resource Cost, and the Total Cost
Begin Optimization With Item 1, Determine Marginal Cost Associated With Each Constraint
C
Compute a Revised Schedule for Item J (a) Modify Cost to Reflect Shadow Price (b) Apply Wagner-Whitin Algorithm
Is this Schedule The Same as Current One?
Yes
No
Is $\hat{C}_j^t \leq 0$?
No
Yes
(1) Compute Revised Resource Requirements (2) Compute Revised Total Cost
Does Revised Schedule Reduce Total System Cost?
No
A
Yes
Adopt Revised Schedule and Record New Costs and Resource Requirements
Determine New Marginal Cost Associated With Each Constraint
Is it Time to Stop?
no
C
A
Yes
End

Flow Diagram of Reflection Programming

Figure IV-1

produce a net improvement in the objective function, the new schedule is adopted. If there is no net improvement, the next item is considered. This process is continued until each of the items has been considered without adopting a new schedule; when this happens, the algorithm stops.

In essence, reflection programming is a heuristic specialization of the simplex method to exploit the particular problem structure considered here.

To see this, recall that the revised simplex method involves the following steps:

1. Obtain an initial, basic feasible solution
2. Determine dual variables associated with the current basis
3. Determine a new column to enter the basis. If a column can be found whose marginal cost C_j is negative, the objective function may be reduced by entering this column into the basis. If there are several columns with $C_j < 0$, the column with the smallest C_j value is selected for basis entry.
4. Determine the variable to leave the basis, and check for an unbounded solution.
5. Update the inverse of the current basis and return to step 2.

Let us now explore how the special structure of Problem II may be exploited in applying each of the steps of the simplex method.

Step 1. Obtain an initial, basic feasible solution (BFS). In this paper we assume that additional resources A_{ik} may always be obtained. Hence, a feasible schedule can be obtained by first computing an initial schedule for each item that satisfies (4)-(9), and then setting the resource variables S_{ik}, O_{ik}, and A_{ik} to provide the required resources. Because of the cost structure assumed here, we would obtain the required resources by first using up any slack, then scheduling overtime up to the maximum of b^o_{ik} hours, and then using auxilary repair sources A_{ik}. With this assumption, a feasible schedule for each item may be obtained by performing the following calculations:

(60) $q^t_{1j} = \max\ (0, d_{1j} - w^*_{1j})$

(61) $q^t_{2j} = \max\ (0, d_{1j} + d_{2j} - w^*_{1j}$

and in general,

(62) $q^t_{ij} = \max\ (0,\ \Sigma^i_{t=1} d_{tj} - w^*_{1j})$

This fancy math corresponds to scheduling repair to just meet net demand. No attempt is made to minimize holding and set-up costs.

Step 2. Determine simplex multipliers.

Let π_{ik}, δ_{ik}, and α_j denote dual variables associated with (17), (18) and (19), respectively. Then the dual of Problem II and the associated primal variables are given by:

(63) Max $\Sigma_{ik}[\pi_{ik} b_{ik} - \delta_{ik} b^o_{ik}] + \Sigma_j \alpha_j$

subject to

(64) $X_j{}^t\!: \Sigma_{ik} \pi_{ik} g^t_{ikj} + \alpha_j \leq P_j{}^t$

(65) $S_{ik}\!: \pi_{ik} \quad \leq R^s_{ik}$

(66) $O_{ik}\!: -\pi_{ik} - \delta_{ik} \quad \leq R^o_{ik}$

(67) $A_{ik}\!: -\pi_{ik} \quad \leq R^A_{ik}$

(68) $\delta_{ik} \quad \geq 0$

From the complementary slackness theorem of linear programming, if a given variable is in the basis, the corresponding dual constraint holds as an equality. Further, the dual variable δ_{ik} corresponds to the overtime constraint (18). Hence $\delta_{ik}=0$ if $O_{ik} < b^o_{ik}$.

We can identify three specific cases associated with each resource. They are:

Case A. If $S_{ik} > 0$, overtime must be zero, which implies $\delta_{ik}=0$. Also from (65),

(69) $\pi_{ik} = R^s_{ik}$

Case B. If $0 < O_{ik} < b^o_{ik}$, we still have $\delta_{ik} = 0$. Hence, (66) must hold as an equality, which implies that:

(71) $\pi_{ik} = -R^o_{ik}$

Case C. If $A_{ik} > 0$, relation (67) must be an equality. Hence,

(72) $\pi_{ik} = -R^A_{ik}$

Relation (66) must also be an equality, since O_{ik} is positive, thus

(73) $\delta_{ik} = R^A_{ik} - R^o_{ik}$

In all three cases above, if π_{ik} and δ_{ik} are known for all ik, and if $x_j{}^t$ is in the basis, then from (64),

(74) $\alpha_j = p_j - \Sigma_{ik}\pi_{ik}g^t_{ijk}$

To summarize, the dual variables associated with a given basis may be established by considering each resource separately. For a given resource, π_{ik} and δ_{ik} are easily determined by applying formulas (69) - (73) as appropriate. Once these values are determined, α_j may be computed using (74).

Step 3.. Determine a new column to enter the basis. First, let us consider the resource variables. Let $\hat{C}^s_{ik}$, $\hat{C}^o_{ik}$, and C^A_{ik} denote the marginal cost associated with the resource variables S_{ik}, O_{ik}, and A_{ik}, respectively. "Pricing out" each of these variables gives us

(75) $\hat{C}^s_{ik} = R^s_{ik} - \pi_{ik}$

(76) $\hat{C}^o_{ik} = R^o_{ik} + \pi_{ik} + \delta_{ik}$

(77) $\hat{C}^A_{ik} = R^A_{ik} + \pi_{ik}$

Suppose there is slack in the basis ($S_{ik} > 0$). Then Case A above holds, giving $\pi_{ik} = R^S_{ik}$ and $\delta_{ik} = 0$. In this case, $\hat{C}^S_{ik} = 0$, and $\hat{C}^O_{ik} > 0$ and $\tilde{C}^A_{ik} > 0$. Hence, if case A holds, increasing either the overtime of subcontracting variable will increase the objective function.

Now suppose Case B above holds, i.e. some overtime is being worked. Then $\delta_{ik} = 0$ and $\pi_{ik} = -R^O_{ik}$. In this case, $C^S_{ik} = R^S_{ik} + R^O_{ik}$, $C^O_{ik} = 0$, and $C^A_{ik} = R^A_{ik} - R^O_{ik}$. Since $R^A_{ik} > R^O_{ik}$, increasing S_{ik} or A_{ik} would increase the objective function.

Finally, suppose Case C holds. Then $\hat{C}^S_{ik} = R^S_{ik} + R^A_{ik}$, $\hat{C}^O_{ik} = R^O_{ik} - R^A_{ik} + R^A_{ik} - R^O_{ik} = 0$ and $\hat{C}^A_{ik} = 0$. Again, the objective function cannot be decreased by increasing one of theses variables.

To summarize, the objective function can never be reduced by bringing one of the resource variables into the basis. Hence, these variables need not be considered as candidates for basis entry at this step.

In Section III, we discussed how a single item dynamic programming problem may be solved to identify a particular schedule with the lowest marginal cost. The same procedure is used in reflection programming. Once the dual variables π_{ik} and α_j are known, a dynamic programming problem may be solved to determine the best candidate schedule Q_j associated with a given item j. If the marginal cost C^t_j associated with this schedule is negative, the algorithm then applies another test: Will the objective function be improved if the current schedule Q_j for item j is replaced by this schedule? This second test is a departure from the simplex method, which will introduce a variable into the basis (perhaps at a fractional level) whenever $C^t_j < 0$. This second test quarantees that the integer nature of reflection programming solution's is retained, but it destroys the quarantee of eventual optimality inherent in the simplex method.

If both tests are satisfied the algorithm proceeds to step 4. Otherwise, another item j is considered. This process of solving item scheduling problems continues until either (a) both tests are passed, or (b) all items have been considered. In the latter case, the algorithm terminates.

Step 4. Determine variable to leave the basis. If a new schedule Q_j^* has been found that passes both tests above, the associated X_j^t is to enter the basis, and the X_j^t associated with the old item schedule is to leave. In making this one-for-one exchange, however, several of the resource variables (S_{ik}, O_{ik}, and A_{ik}) may be affected. However, the required adjustments are easily determined by computing the differences in resource requirements associated with the old and new schedules.

Step 5. Update the inverse of the current basis.

As we have seen all of the above steps may be performed without reference to the inverse of the current basis. Hence, there is no need to store nor to update the inverse in applying reflection programming. This is one of the major computational advantages of the method.

Section V. Computation Experience

To evaluate the potential usefulness of Reflection Programming (RFLP), we performed two types of computational experiments. In the first computational experiments, we compared RFLP results with solutions obtained using independent item assumptions. Our second experiment compared RFLP results with those obtained using Generalized Upper Bounding (GUB).

RFLP/Indpendent Item Comparison

To test the characteristics of reflection programming, we formulated a scenario involving 4 resources, 12 planning periods, and 170 items for the Warmdot Company, a hypothetical manufacturing firm studied extensively by Brown (1967). When stated as an integer optimization problem in the form of (1)-(9), the resulting problem has 266 constraints and over 340,000 columns--an optimization problem with substantial dimensions. The data for this scenario was constructed by a Monte Carlo Process to have the same statistical characteristics as the top 170 items in the Warmdot inventory. Four basic data sets were generated, each with differing degrees of seasonality as outlined in Table V-1.

Reflection programming was applied to each of the data sets under two different cost structures; hence a total of eight runs was performed. In the first cost structure, it was assumed that:

Cost per hour slack = \$4.00

Cost per hour overtime = \$6.00

Cost per hour subcontracted = \$60.00

Cost per period for holding inventory = 2% of unit cost

Set-up cost = \$1.00

The second cost structure was identical to the above, with the exception that the cost of slack was set to zero.

In all tests, an initial schedule was obtained by applying the Wagner-Whitin algorithm to each item independently. Reflection programming was then applied in an attempt to improve the initial schedules.

Results of our experimental runs are shown in Table V-2. These runs demonstrate the speed of the reflection programming. In all eight runs, it took 90 seconds or less of CPU time of IBM 360/65 to reach a heuristic solution. When the demand rate is constant, the cost reduction from the initial solution to the final solution were either moderate (19%) or insignificant (1.3%). However, when seasonality is present, the cost reduction was substantial, ranging from 82% to 97%. These results are quite encouraging, since significant cost reductions were obtained from very modest expenditures of computer time.

Table V-1
Four Types of Demand Rate

Demand Rate	Distribution
Constant Rate	Monthly demand is normally distributed at a constant rate through the year
Mild Seasonal	Maximum monthly demand = 20% of annual demand
	Minimum monthly demand = 2% of annual demand
Highly Seasonal	Maximum monthly demand = 30% of annual demand
	Minimum monthly demand = 1% of annual demand
Extreme Seasonal	Range from 3 months of "0" demand up to 25% per month of the annual demand

Table V-2

Summary of Experimental Runs

Statistics	Constant Demand Rate		Mild Seasonal Rate		Highly Seasonal Rate		Extremely Seasonal	
	Cost of Slack		Cost of Slack		Cost of Slack		Cost of Slack	
	$4.00	Not Cost	$4.00	No Cost	$4.00	Not Cost	$4.00	Not Cost
Initial Solution	$131,352	$ 5,085	$3,867,611	$2,507,409	$6,298,966	$5,562,524	$3,738,943	$2,911,761
Final Solution	129,689	4,120	358,505	84,141	697,868	288,630	677,430	76,598
Number of Passes Through Each Item	7	5	146	140	144	139	92	92
Total CPU Time (in seconds)	46	30	56	90	57	52	43	72
Percent of Cost Reduction from Initial Solution to Final Solution	1.3%	19%	91%	97%	89%	95%	82%	97%

RFLP/GUB Comparisons

Originally, we intended to compare the RFLP and GUB algorithms in the same senario as described above. However, our GUB algorithm has proven to be quite inefficient (and extremely expensive.). It appears that the inefficiency is due to the way we store and retrieve data in the algorithm, rather than a problem with the algorithm itself, but we are still not certain as to the exact problem. Hence, our comparisons have been restricted to much smaller problems.

As illustrated in Table V-3, four problems were solved using the RFLP and GUB algorithms. In all cases, GUB produced a lower cost solution, but required from 3 to 10 times as much CPU time. For these small problems the reduction in schedule cost produced by GUB easily pays for the increased computer costs. For larger problems, however, this may not be true. For example, in attempting to solve the 4 resource, 12 period, 170 item problem discussed above, our GUB algorithm run was terminated at 1200 seconds of CPU time. In contrast, RFLP reached it's final solution in 90 seconds for this problem, and the final RFLP solution was better than the GUB solution at the time of termination.

Unfortunately, our GUB code appears to be very inefficient, so that it is difficult at this time to conclude which method would be most efficient.

Table V-3. Comparison of RFLP And GUB Solutions

Resources	Periods	Items	Method	Solution Total Cost	CPU sec	Computer Job Change $
1	6	50	RFLP	6,608	10	$ 1.25
			GUB	4,492	65	$ 5.50
1	12	50	RFLP	5,795	16	$ 1.80
			GUB	4,570	169	$10.31
2	4	10	RFLP	1,161	4	$ .75
			GUB	367	14	$ 1.35
2	4	20	RFLP	3,852	5	$ .89
			GUB	2,772	24	$ 2.06

Agin, N.I.,and D.E. Cullen, "An Algorithm for Transportation Routing and Vehicle Loading," Studies In Management Sciences, Vol. I, Logistics, M.A. Geisler (ed.), North Holland, 1972, pp. 1-20.

Allen, S.G., and D.A. D'Esopo, "An Ordering Policy for Repairable Stock Items," Operations Research, v16, n3, May-June 1968, pp. 669-674.

Balinski, M.L., "Fixed Cost Transportation Problem," Naval Research Logistics Quarterly, Vol.8, January, 1961 pp. 41-54.

Brown, G.F., T.M. Corcoran, and R.M. Lloyd, A Dynamic Inventory Model with Delivery Lag & Repair, Center for Naval Analyses, University of Rochester, Aug. 1, 1969.

Cooper, L., and C. Drebes, "An Approximate Solution for the Fixed Charge Problems," Naval Research Logistics Quarterly, Vol. 14, No. 1, March, 1967, pp. 101-113.

Demmy, W.S., Allocation of Spares and Repair Resources to a Multi Component System, AFLC Report 70-17, Office of DCS/Comptroller, Headquarters Air Force Logistics Command, Wright-Patterson Air Force Base, Ohio.

Demmy, W.S., "Computing Multi-Item Procurement and Repair Schedules in Reparable-Item Inventory Systems," Working Paper, Ernst and Ernst, 2300 Winters Bank Building, Dayton, Ohio (1974).

Denzler, D.R., "An Approximate Algorithm for the Fixed Charge Problem," Naval Research Logistics Quarterly, Vol. 16, No. 3, September 1969, pp. 411-258.

Dzielinski, B.P., C.T. Baker, and A.S. Manne, Simulation Tests of Lot Size Programming, Management Schience, v9, n2, January 1963, pp. 229-258.

Dzielinski, B.P. and R.E. Gomory, Optimum Programming of Lot Sizes Inventories, and Labor Allocation, IBM- ASDD TR 17-120, May, 1964.

Dzielinski, B.P. and R.E. Gomory, "Optimum Programming of Lot Sizes, Inventories, and Labor Allocations," Management Science, v11, n9, July 1965, pp. 874-890.

Fawcett, W.M. and R.D. Gilbert, "A Non-Steady State Stochastic Representation of a Supply System for Aircraft Spares," ERR-FW-1349, General Dynamics Convair Division, 29 December 1972, p. 52.

Fawcett, W.M. and L.J. York, "Base-Depot Stockage Model," Research Report ERR-FW-621, General Dynamics Dort Worth Division, 29 Dec. 1967, p. 83.

Freiheit, James E., A Continuous Review Model for the Reparable Item Inventory System, master's thesis, Naval Postgraduate School, Monterey, California, June 1967, p. 47 (AD 817 560).

Gorenstein, S., "Planning Tire Production," Management Science, Vol. 17, No. 2, October, 1970, pp. B72-B82.

Gray, P., "Exact Solution of the Fixed-Charge Transportation Problem," Operations Research, Vol. 19, October, 1971, pp. 1529-1538.

Hastings, David A., A Constrained Periodic Review Model for a Probabilistic Reparable-Item Inventory System, Master's Thesis, U.S. Naval Postgraduate School, Monterey, California, June 1967, p. 31 (AD 823 751).

Jones, A.P., and R.M. Soland, " A Branch-and Bound Algorithm for Multi-Level Fixed Charge Problems," Management Science, Vol. 16, No. 1, September, 1969, pp. 67-76.

Kochman, G.A., "On a Class of Concave-Separable Interger Programs," Technical Report, Department of Operations Research, Stanford University, September 1976.

Kortanek, L.O., D. Sodaro and A.L. Soyster, "Multi-Product Production Scheduling Via Extreme Point Properties of Linear Programming," Naval Research Logistics Quarterly, v15, n2, June, 1968, pp. 287-300.

Krause, W.K., Optimal Multi-Echelon Stockage Policies with Constraints, Inventory Research Office, United States Army Logistics Management Center, Fort Lee, Va., Feb. 1970, p. 52.

Lasdon, L.S., and R.C. Terjung, "An Efficient Algorithm for Multi-Item Scheduling," Operations Research, Vol. 19, No. 4, July-August, 1971, pp. 946-969.

Manne, A.S., "Programming of Economic Lot Sizes," Management Science, v4, n2, January, 1958, pp. 115-135.

Murty, K.G., "Solving the Fixed Charge Problem by Routing Extreme Points," Operations Research, Vol. 16, No. 2, March-April, 1968, pp. 268-279.

Muckstadt, John A., "A Model for a Multi-Item, Multi-Echelon, Multi-Indenture Inventory System, Management Science, v20, n4, December, 1973, pp. 472-481.

Porteus, E. and Z. Lansdowne, "Optimal Design of a Multi-Item Multi-Location Multi-Repair Type Repair and Supply System," Nav. Res. Log. Quart. 21, 213-238 (197).

Rousseau, J.M., "A Cutting Plane Method for the Fixed Cost Problem", unpublished doctoral thesis, Sloan School of Management, MIT, August, 1973.

Schrady, D.A., "A Deterministic Inventory Midel for Reparable Items," Naval Research Logistics Quarterly, v. 14, n. 3, Sept. 1967, pp. 397-398.

Schrady, David A., Mathematical Models of the Reparable Item Inventory System, Naval Postgradate School, Monterey, California, NPS 55SO7081A, August 1967, 145 p. (AD 660 371).

Sherbrooke, C.C., METRIC: A Multi-Echelon Technique for Recoverable Item Control, The RAND Corporation, RM-5078-PR, November 1966. Also in Operations Research, v. 16, pp. 122-141, 1968.

Sherbrooke, C.C., MINE: Multi-Identure NORS Evaluation, RM-5826-PR, The RAND Corp., Dec. 1968, 35 pp.

Simon, Richard M., Stationary Properties of a Two-Echelon Inventory Model for Low Demand Items, The RAND Corp., RM-5928-PR, May 1969, 37 pp.

Steinberg, D.I., "The Fixed Charge Problem", Naval Research Logistics Quarterly, Vol. 17, No. 2, June, 1970, pp. 217-235.

Wagner, H.M. and T.M. Whitin, "A Dynamic Version of the Economic Lot Size Model," Management Science, V. 5, 1958, pp. 89-96.

SECURITY CLASSIFIC[illegible]OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 77-1293	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) MULTI-ITEM SCHEDULING IN REPARABLE ITEM INVENTORY SYSTEMS WITH REFLECTION PROGRAMMING		5. TYPE OF REPORT & PERIOD COVERED Interim
		6. PERFORMING ORG. REPORT NUMBER WP-76-3011-4
7. AUTHOR(s) W. Steven Demmy		8. CONTRACT OR GRANT NUMBER(s) AFOSR 76-3011
9. PERFORMING ORGANIZATION NAME AND ADDRESS Wright State University Dept of Administrative Science & Finance Dayton, OH 45435		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2304/A5
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Office of Scientific Research/NM Bolling AFB, DC 20332		12. REPORT DATE April 1977
		13. NUMBER OF PAGES 33
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

multi-item, scheduling, reparable, inventory, reflection programming

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The repair and overhaul of military equipment is a costly and complex logistics activity. Factors that may have significant impact upon scheduling decisions in such systems include costs of set-up, production, and expediting, obsolescence probabilities, the availability of the reparable assets, manpower, equipment and funds required to support the repair effort. This paper reviews the current state-of-the-art for solving multi-item repair scheduling problems in such systems, and presents an algorithm for solving these problems using — next page

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. Abstract

Generalized Upper Bounding with dynamic programming subproblems. Computational experience involving up to 170 items, 48 constraints, and 12 planning periods is also reported.